Chapter 10

Error Detection and Correction
检错与纠错

Note

Data can be corrupted during transmission.

Some applications require that errors be detected and corrected.

数据在出传输额过程中可能遭到破坏，一些应用需要进行检错和纠错。

10-1 INTRODUCTION

Let us first discuss some issues related, directly or indirectly, to error detection and correction.

首先直接或间接地讨论有关差错检测和差错纠正的问题

Topics discussed in this section:

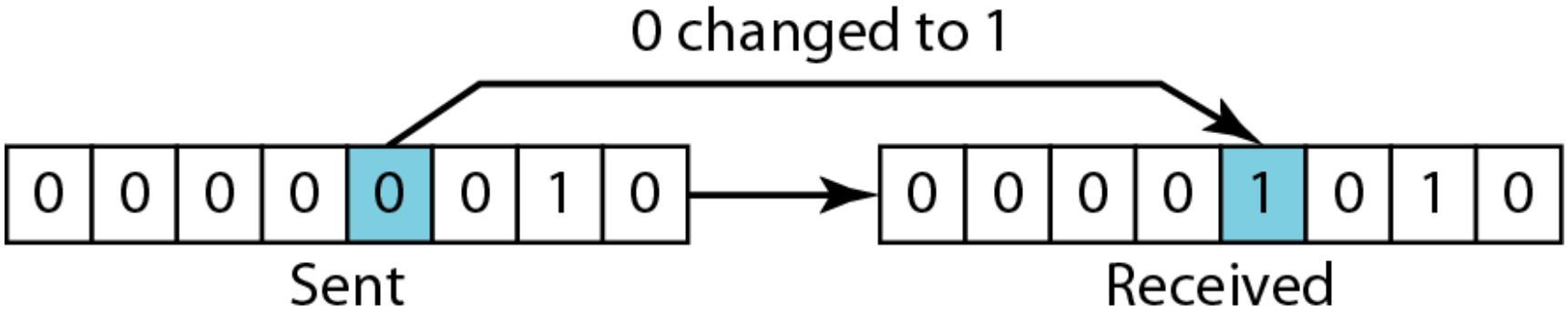
Types of Errors 差错类型
Redundancy 冗余
Detection Versus Correction 检错和纠错
Forward Error Correction Versus Retransmission 前向纠错和重传
Coding 编码
Modular Arithmetic 模运算

Note

In a single-bit error, only 1 bit in the data unit has changed.

在单比特差错中，数据单位中仅有一比特发生变化。

Figure 10.1 *Single-bit error*

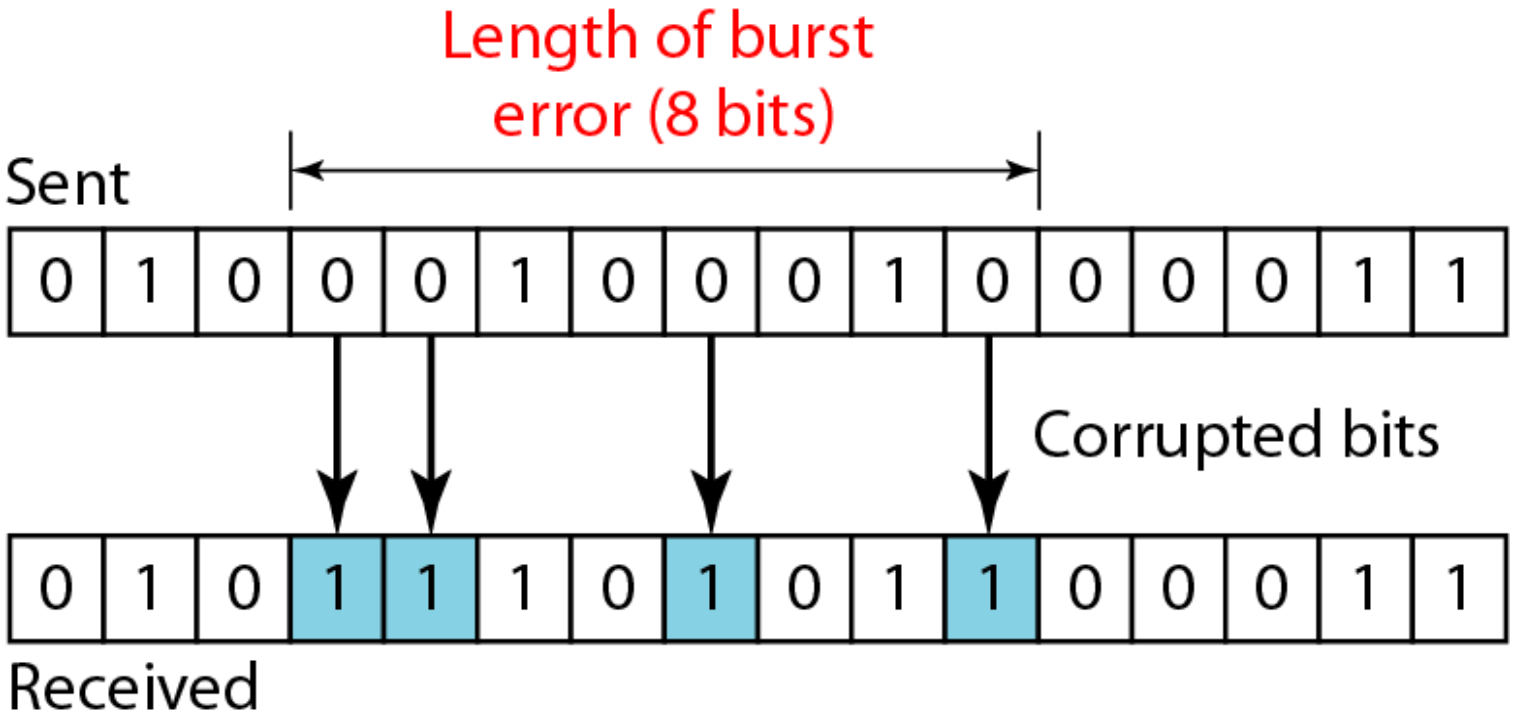


Note

A burst error means that 2 or more bits in the data unit have changed.

一个突发差错意味着数据单元中两位或多位发生变化。

Figure 10.2 *Burst error of length 8*

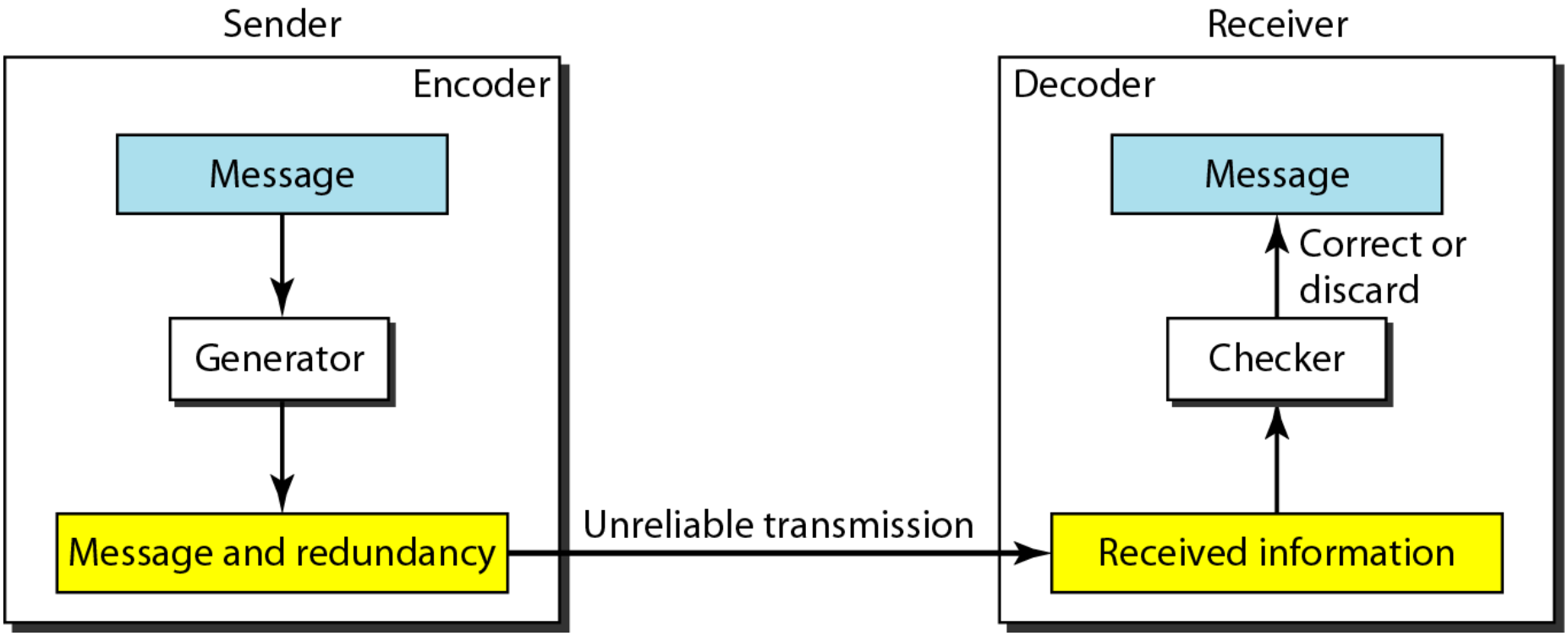


Note

To detect or correct errors, we need to send extra (redundant) bits with data.

为了检测或纠正错误，我们需要发送除了数据外的额外（冗余）位。

Figure 10.3 *The structure of encoder and decoder 编码器和译码器的结构*



Note

In this book, we concentrate on block codes; we leave convolution codes to advanced texts.

本课程，我们主要介绍块编码。

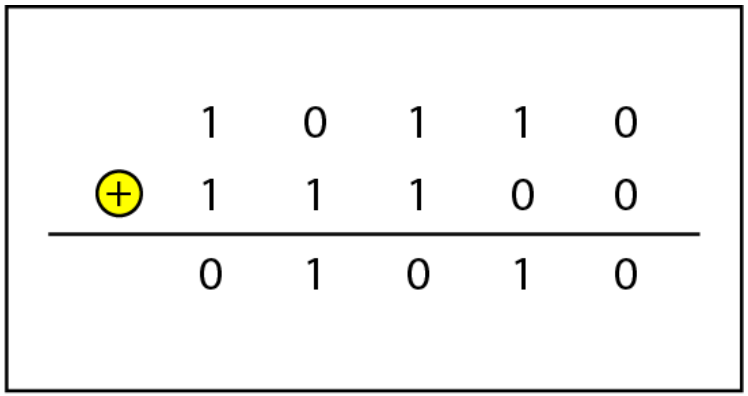
Note

In modulo-N arithmetic, we use only the integers in the range 0 to N −1, inclusive.
在模N运算中，只使用0到N-1的整数

Figure 10.4 *XORing of two single bits or two words 模2运算和异或运算关系*

a. Two bits are the same, the result is 0.

b. Two bits are different, the result is 1.



c. Result of XORing two patterns

10-2 BLOCK CODING 块编码

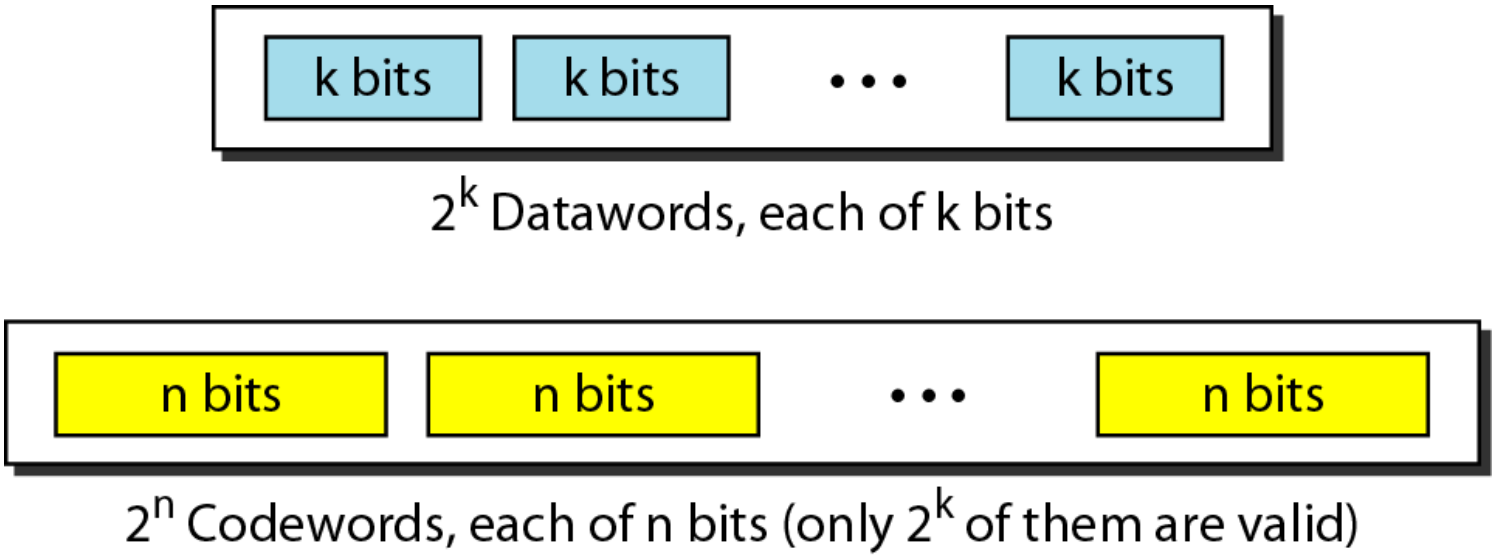
In block coding, we divide our message into blocks, each of k bits, called datawords. We add r redundant bits to each block to make the length n = k + r. The resulting n-bit blocks are called codewords.

在块编码中，我们把报文划分成块，每块k位，称数据字，并增加r个冗余位使其长度变为n=k+r，形成n位块称为码字。

Topics discussed in this section:

Error Detection 检错
Error Correction 纠错
Hamming Distance 汉明距离
Minimum Hamming Distance 最小汉明距离

Figure 10.5 *Datawords and codewords in block coding 数据字和码字*

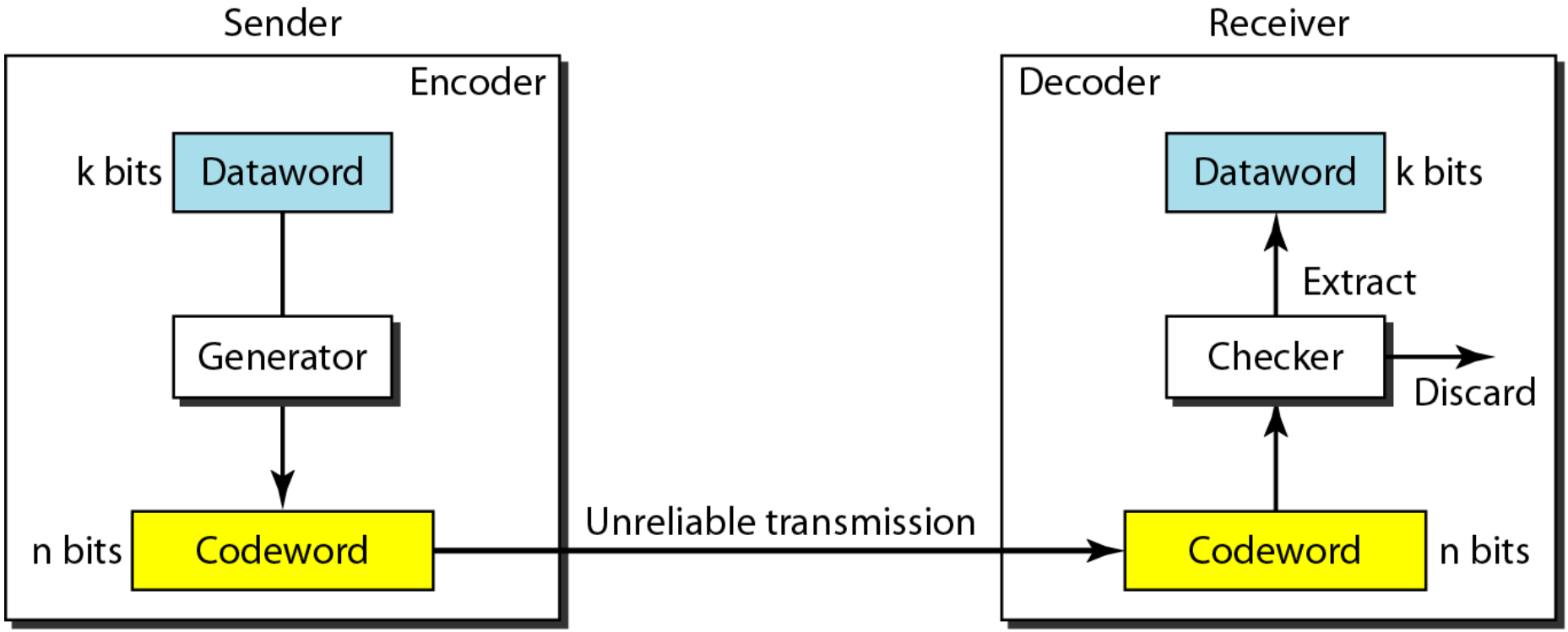


Example 10.1

The 4B/5B block coding discussed in Chapter 4 is a good example of this type of coding. In this coding scheme, k = 4 and n = 5. As we saw, we have $2^k = 16$ datawords and $2^n = 32$ codewords. We saw that 16 out of 32 codewords are used for message transfer and the rest are either used for other purposes or unused.

第四章讨论的4B/5B码就是这种编码的一个好例子。我们可以看到，由于k=4，n=5，则数据字有$2^k = 16$ 个，码字有$2^n = 32$ 个。我们从32个码字里精心挑选16个码字用于报文传输。其余16个不用。

Figure 10.6 *Process of error detection in block coding块编码的差错检测过程*



Example 10.2

只能检测一比特的检错码例子

Let us assume that k = 2 and n = 3. Table 10.1 shows the list of datawords and codewords. Later, we will see how to derive a codeword from a dataword.

Assume the sender encodes the dataword 01 as 011 and sends it to the receiver. Consider the following cases:

1. *The receiver receives 011. It is a valid codeword. The receiver extracts the dataword 01 from it.*
2. *The codeword is corrupted during transmission, and 111 is received. This is not a valid codeword and is discarded.*
3. *The codeword is corrupted during transmission, and 000 is received. This is a valid codeword. The receiver incorrectly extracts the dataword 00. Two corrupted bits have made the error undetectable.*

Table 10.1 ***A code for error detection (Example 10.2)***

Datawords	*Codewords*
00	000
01	011
10	101
11	110

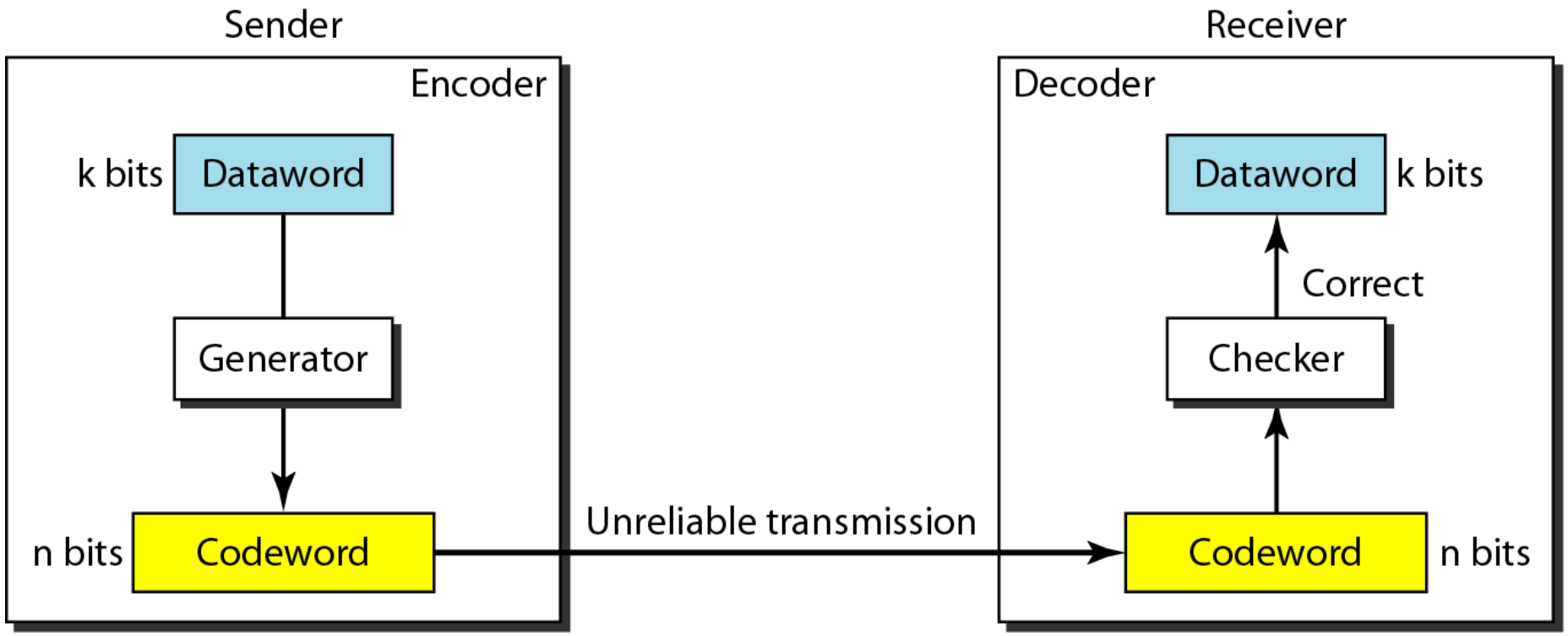
Note

An error-detecting code can detect only the types of errors for which it is designed; other types of errors may remain undetected.

检错码是为某些类型的差错而设计的，因此只能检测这些类型的差错；其它类型的差错就无法检测到。

Figure 10.7 ***Structure of encoder and decoder in error correction***

纠错码的编码器和译码器的结构



Example 10.3

只能纠一比特错误的纠错码例子

Let us add more redundant bits to Example 10.2 to see if the receiver can correct an error without knowing what was actually sent. We add 3 redundant bits to the 2-bit dataword to make 5-bit codewords. Table 10.2 shows the datawords and codewords. Assume the dataword is 01. The sender creates the codeword 01011. The codeword is corrupted during transmission, and 01001 is received. First, the receiver finds that the received codeword is not in the table. This means an error has occurred. The receiver, assuming that there is only 1 bit corrupted, uses the following strategy to guess the correct dataword.

Example 10.3 (continued)

1. *Comparing the received codeword with the first codeword in the table (01001 versus 00000), the receiver decides that the first codeword is not the one that was sent because there are two different bits.*

2. *By the same reasoning, the original codeword cannot be the third or fourth one in the table.*

3. *The original codeword must be the second one in the table because this is the only one that differs from the received codeword by 1 bit. The receiver replaces 01001 with 01011 and consults the table to find the dataword 01.*

Table 10.2 ***A code for error correction (Example 10.3)***

Dataword	*Codeword*
00	00000
01	01011
10	10101
11	11110

Note

The Hamming distance between two words is the number of differences between corresponding bits.

两个字的汉明距离是对应位不同的数量。

Example 10.4

用异或计算汉明距离

Let us find the Hamming distance between two pairs of words.

1. The Hamming distance d(000, 011) is 2 because

$000 \oplus 011$ is 011 (two 1s)

2. The Hamming distance d(10101, 11110) is 3 because

$10101 \oplus 11110$ is 01011 (three 1s)

Note

The minimum Hamming distance is the smallest Hamming distance between all possible pairs in a set of words.

最小汉明距离是一组字中所有可能对的最小汉明距离。

Example 10.5

Find the minimum Hamming distance of the coding scheme in Table 10.1.

求表10.1编码方案的最小汉明距离。

Solution

We first find all Hamming distances.

$d(000, 011) = 2$ $d(000, 101) = 2$ $d(000, 110) = 2$ $d(011, 101) = 2$
$d(011, 110) = 2$ $d(101, 110) = 2$

The d_{min} in this case is 2.

Datawords	*Codewords*
00	000
01	011
10	101
11	110

Example 10.6

Find the minimum Hamming distance of the coding scheme in Table 10.2.

求表10.2编码方案的最小汉明距离。

Solution

We first find all the Hamming distances.

$d(00000, 01011) = 3$	$d(00000, 10101) = 3$	$d(00000, 11110) = 4$
$d(01011, 10101) = 4$	$d(01011, 11110) = 3$	$d(10101, 11110) = 3$

The d_{min} in this case is 3.

Dataword	*Codeword*
00	00000
01	01011
10	10101
11	11110

Note

To guarantee the detection of up to s errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = s + 1$.

为了保证检测出最多s个错误，块编码中最小汉明距离一定是$d_{min} = s + 1$。

Example 10.7

表10.1第一个编码方案的最小汉明距离为1，所以只能检单比特错误。

The minimum Hamming distance for our first code scheme (Table 10.1) is 2. This code guarantees detection of only a single error. For example, if the third codeword (101) is sent and one error occurs, the received codeword does not match any valid codeword. If two errors occur, however, the received codeword may match a valid codeword and the errors are not detected.

Datawords	*Codewords*
00	000
01	011
10	101
11	110

Example 10.8

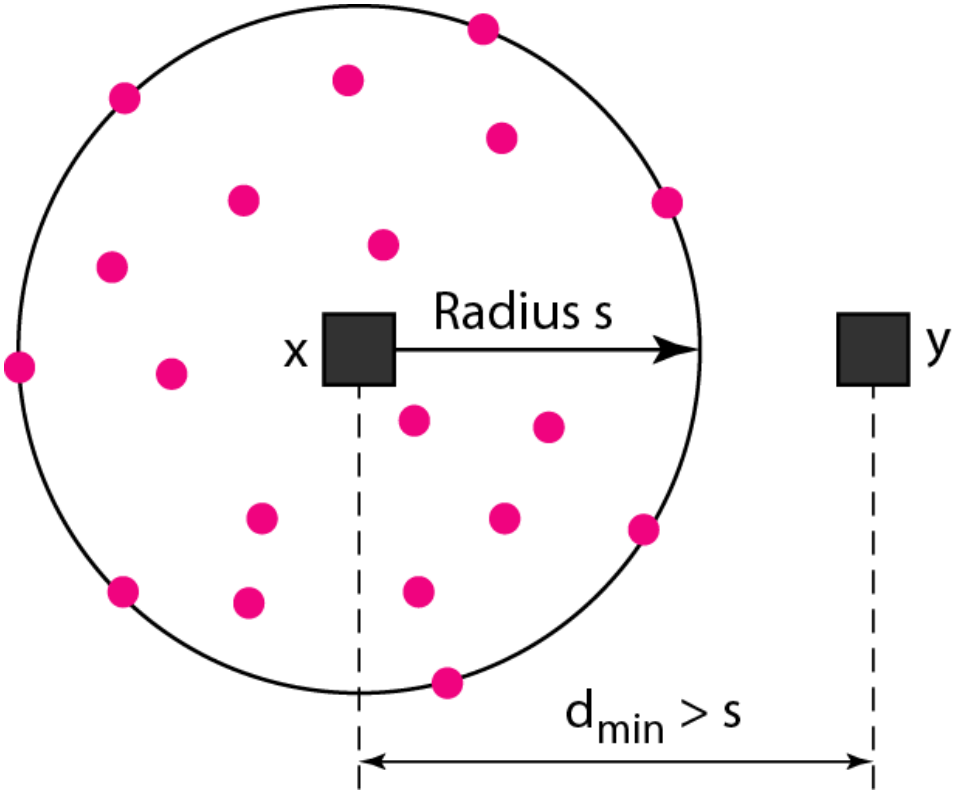
表10.2第二个编码方案的最小汉明距离为3，所以最多能检两比特差错。

Our second block code scheme (Table 10.2) has $d_{min} = 3$. This code can detect up to two errors. Again, we see that when any of the valid codewords is sent, two errors create a codeword which is not in the table of valid codewords. The receiver cannot be fooled.

However, some combinations of three errors change a valid codeword to anothe accepts the received c undetected.

Dataword	*Codeword*
00	00000
01	01011
10	10101
11	11110

Figure 10.8 *Geometric concept for finding d_{min} in error detection* 几何意义



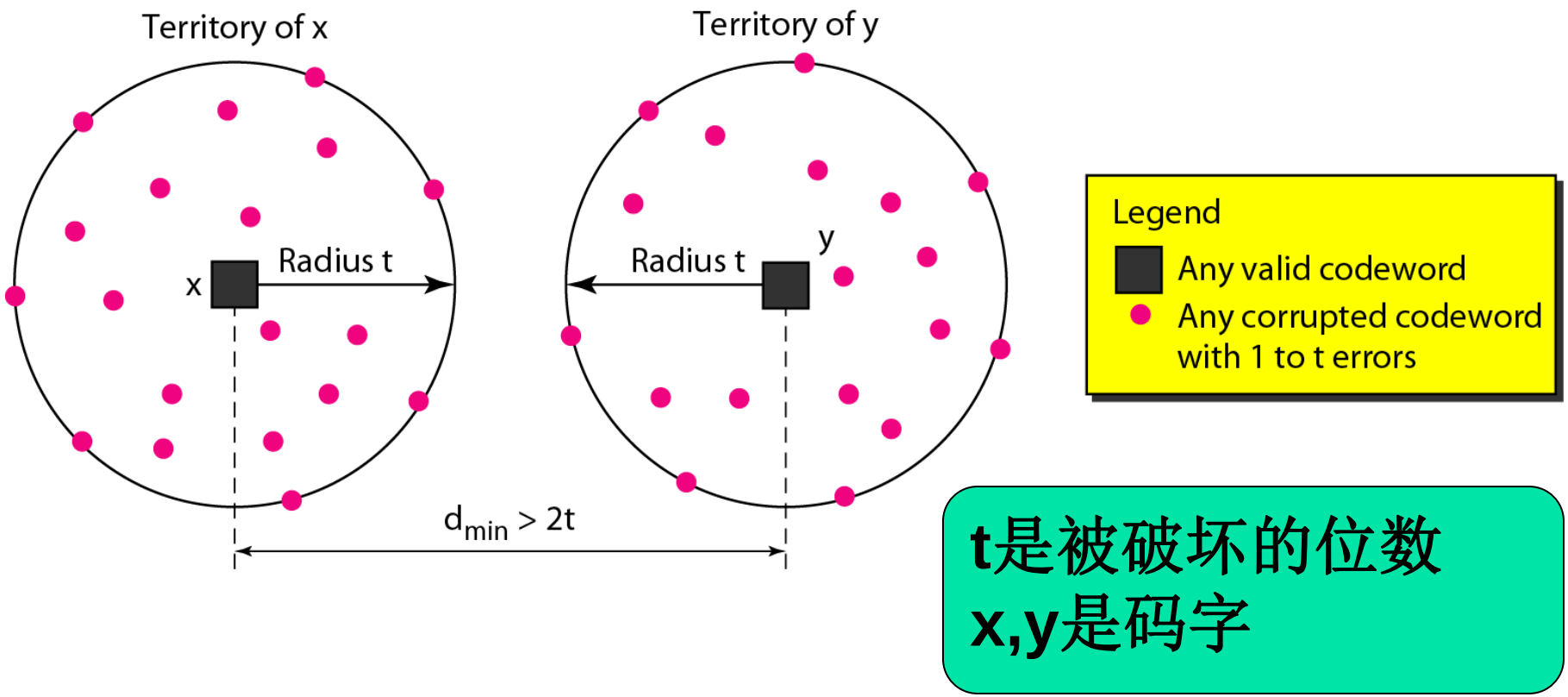
Legend

Any valid codeword

Any corrupted codeword with 0 to s errors

s是差错的位数
x是码字

Figure 10.9 *Geometric concept for finding d_{min} in error correction*
纠错码几何意义



t是被破坏的位数
x,y是码字

Note

To guarantee correction of up to *t* errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = 2t + 1$.

为了保证最多能纠正*t*个差错，块码中最小汉明距离是$d_{min} = 2t + 1$。

Example 10.9

A code scheme has a Hamming distance d_{min} = 4. What is the error detection and correction capability of this scheme?
汉明距离为4的编码方案，检错和纠错能力分别是多少？

Solution

This code guarantees the detection of up to three errors (s = 3), but it can correct up to one error. In other words, if this code is used for error correction, part of its capability is wasted. 纠错码需要的最小距离是奇数 (3, 5, 7, . . .).

10-3 LINEAR BLOCK CODES

Almost all block codes used today belong to a subset called linear block codes. A linear block code is a code in which the exclusive OR (addition modulo-2) of two valid codewords creates another valid codeword.

当前，几乎所有使用的块码都属于一个称为线性块编码的子集。在线性快码中，任两个有效码字的异或（即模二加）生成另一个有效码字。（需要抽象代数里有限域的概念）

Topics discussed in this section:

Minimum Distance for Linear Block Codes 线性块码的最小距离
Some Linear Block Codes 一些线性块编码

Note

In a linear block code, the exclusive OR (XOR) of any two valid codewords creates another valid codeword.

在线性快码中，任两个有效码字的异或（即模二加）生成另一个有效码字。

Example 10.10

表10.1和10.2都属于线性块编码。

Let us see if the two codes we defined in Table 10.1 and Table 10.2 belong to the class of linear block codes.

1. *The scheme in Table 10.1 is a linear block code because the result of XORing any codeword with any other codeword is a valid codeword. For example, the XORing of the second and third codewords creates the fourth one.*

2. *The scheme in Table 10.2 is also a linear block code. We can create all four codewords by XORing two other codewords.*

Example 10.11

最小汉明距离求法：是具有最小1的个数的非0有效码字中1的个数。

In our first code (Table 10.1), the numbers of 1s in the nonzero codewords are 2, 2, and 2. So the minimum Hamming distance is $d_{min} = 2$. *In our second code (Table 10.2), the numbers of 1s in the nonzero codewords are 3, 3, and 4. So in this code we have* $d_{min} = 3$.

Note

A simple parity-check code is a single-bit error-detecting code in which $n = k + 1$ with $d_{min} = 2$.

简单的奇偶校验码是$n = k + 1$，且 $d_{min} = 2$ 的单比特检错码。

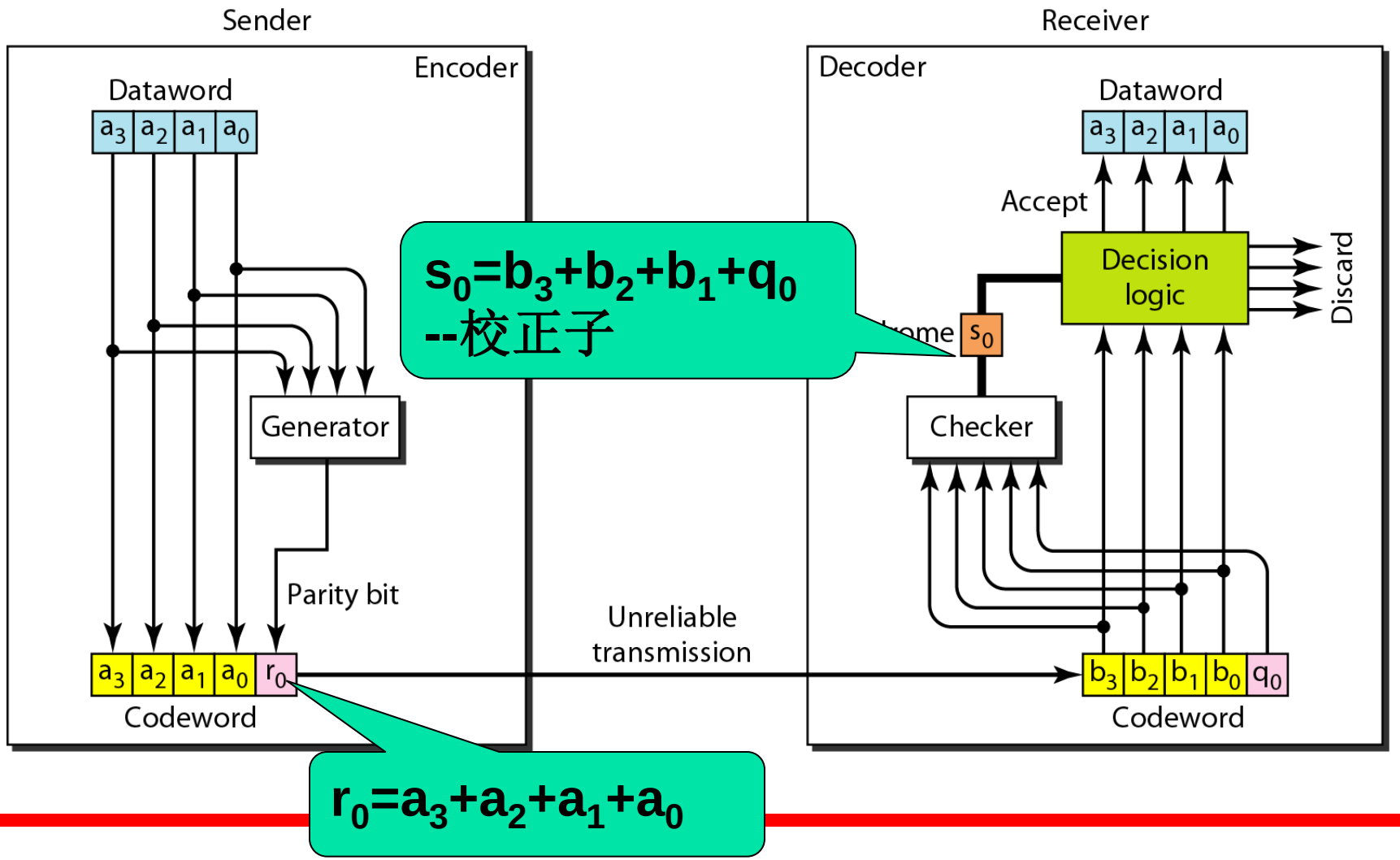
Table 10.3 *Simple parity-check code C(5, 4) 简单的奇偶校验码*

Datawords	*Codewords*	*Datawords*	*Codewords*
0000	00000	1000	10001
0001	00011	1001	10010
0010	00101	1010	10100
0011	00110	1011	10111
0100	01001	1100	11000
0101	01010	1101	11011
0110	01100	1110	11101
0111	01111	1111	11110

编码方案写成C(n,k)和一个单独的d_{min}表达式
n是码字的长度，k是数据的位数， d_{min}是最小汉明距离

Figure 10.10 *Encoder and decoder for simple parity-check code*

奇偶校验码的编码器和译码器 （模二加运算）



Example 10.12

奇偶校验只能检出奇数个错误，不能检出偶数个错误

Let us look at some transmission scenarios. Assume the sender sends the dataword 1011. The codeword created from this dataword is 10111, which is sent to the receiver. We examine five cases:

1. *No error occurs; the received codeword is 10111. The syndrome(校正子) is 0. The dataword 1011 is created.*
2. *One single-bit error changes a_1. The received codeword is 10011. The syndrome is 1. No dataword is created.*
3. *One single-bit error changes r_0. The received codeword is 10110. The syndrome is 1. No dataword is created.*

Example 10.12 (continued)

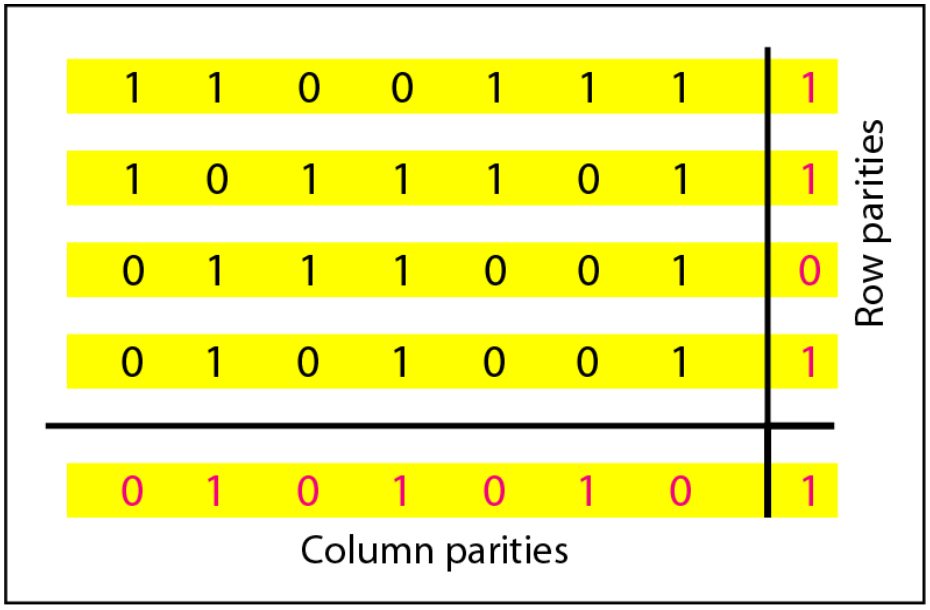
4. An error changes r_0 and a second error changes a_3. The received codeword is 00110. The syndrome is 0. The dataword 0011 is created at the receiver. Note that here the dataword is wrongly created due to the syndrome value.
5. Three bits—a_3, a_2, and a_1—are changed by errors. The received codeword is 01011. The syndrome is 1. The dataword is not created. This shows that the simple parity check, guaranteed to detect one single error, can also find any odd number of errors.

Note

A simple parity-check code can detect an odd number of errors.

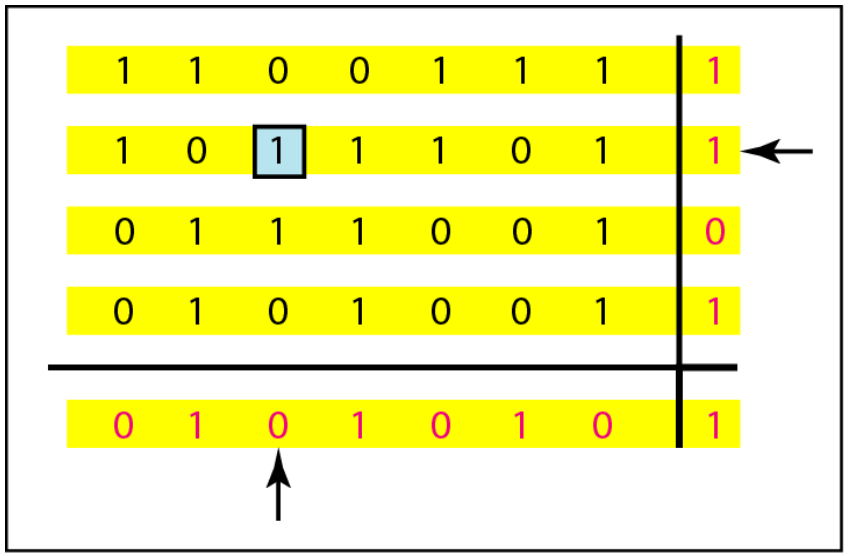
简单奇偶校验码能检出奇数个差错。

Figure 10.11 *Two-dimensional parity-check code*

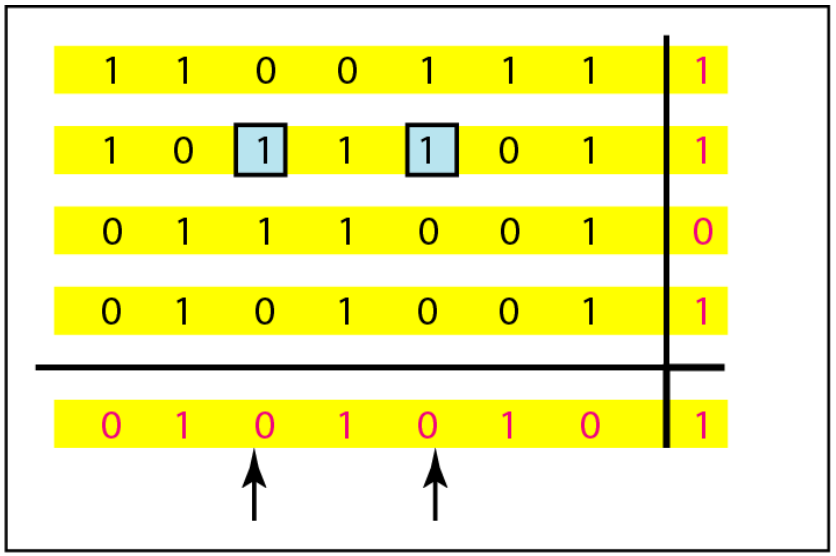


a. Design of row and column parities

Figure 10.11 *Two-dimensional parity-check code*

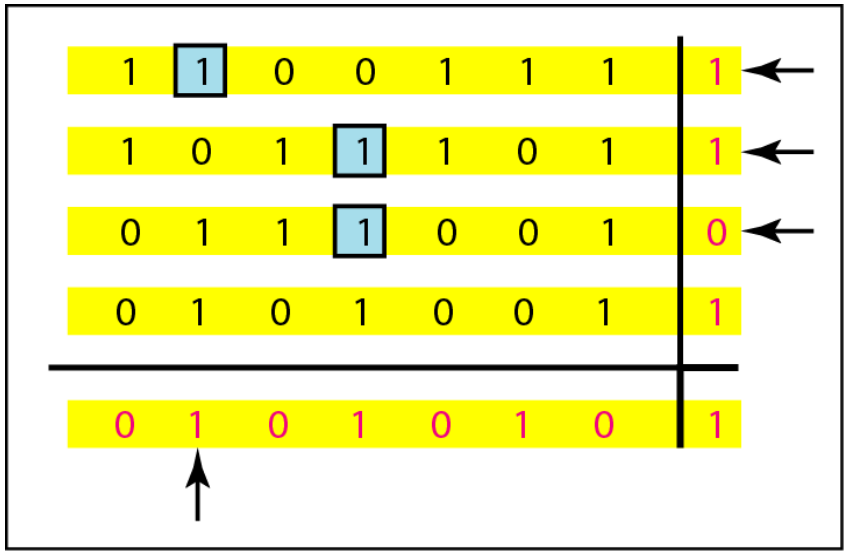


b. One error affects two parities

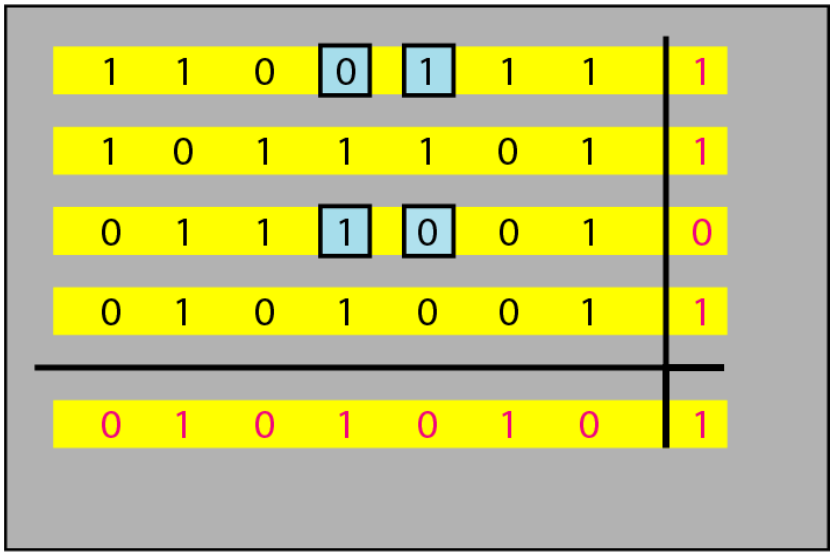


c. Two errors affect two parities

Figure 10.11 *Two-dimensional parity-check code*



d. Three errors affect four parities



e. Four errors cannot be detected

- 两维奇偶校验能检测出表中任何位置发生的最多三个差错
- 4位差错无法检出

Note

All Hamming codes discussed in this book have $d_{min} = 3$.

本书只讨论汉明距离为3的最小的汉明码，这样能检出2位差错，和纠正1位差错。

The relationship between *m* and *n* in these codes is *n* = 2^*m* − 1

汉明距离m与码字长n和数据字长k的关系为

n* = 2^*m* − 1, *k* = *n*-*m*, *r* = *n* - *k

Table 10.4 ***Hamming code C(7, 4)***

Datawords	*Codewords*	*Datawords*	*Codewords*
0000	0000000	1000	1000110
0001	0001101	1001	1001011
0010	0010111	1010	1010001
0011	0011010	1011	1011100
0100	0100011	1100	1100101
0101	0101110	1101	1101000
0110	0110100	1110	1110010
0111	0111001	1111	1111111

Figure 10.12 *The structure of the encoder and decoder for a Hamming code*

汉明码的编码器和解码器的结构（冗余位也是模2运算，见书）

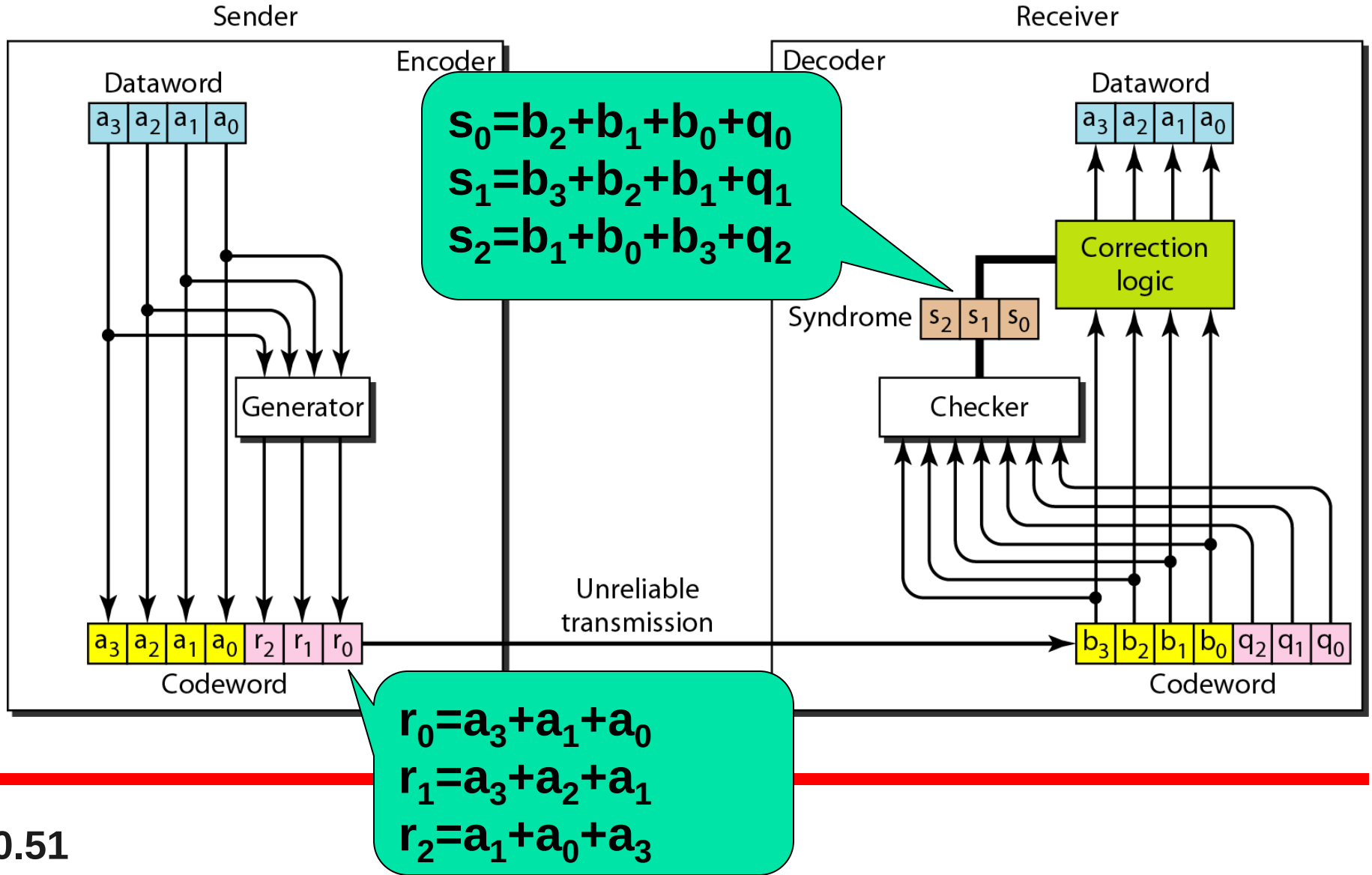


Table 10.5 ***Logical decision made by the correction logic analyzer***
译码器的纠错逻辑分析器的逻辑判定

Syndrome	000	001	010	011	100	101	110	111
Error	None	q_0	q_1	b_2	q_2	b_0	b_3	b_1

Example 10.13

Let us trace the path of three datawords from the sender to the destination:

跟踪3个数据字从发送端到目的端的路径，见书P185

1. *The dataword 0100 becomes the codeword 0100011. The codeword 0100011 is received. The syndrome is 000, the final dataword is 0100.*
2. *The dataword 0111 becomes the codeword 0111001. The syndrome is 011. After flipping b_2 (changing the 1 to 0), the final dataword is 0111.*
3. *The dataword 1101 becomes the codeword 1101000. The syndrome is 101. After flipping b_0, we get 0000, the wrong dataword. This shows that our code cannot correct two errors.*

Example 10.14

We need a dataword of at least 7 bits. Calculate values of k and n that satisfy this requirement.

若数据字至少为7比特，计算n，k，r。

Solution

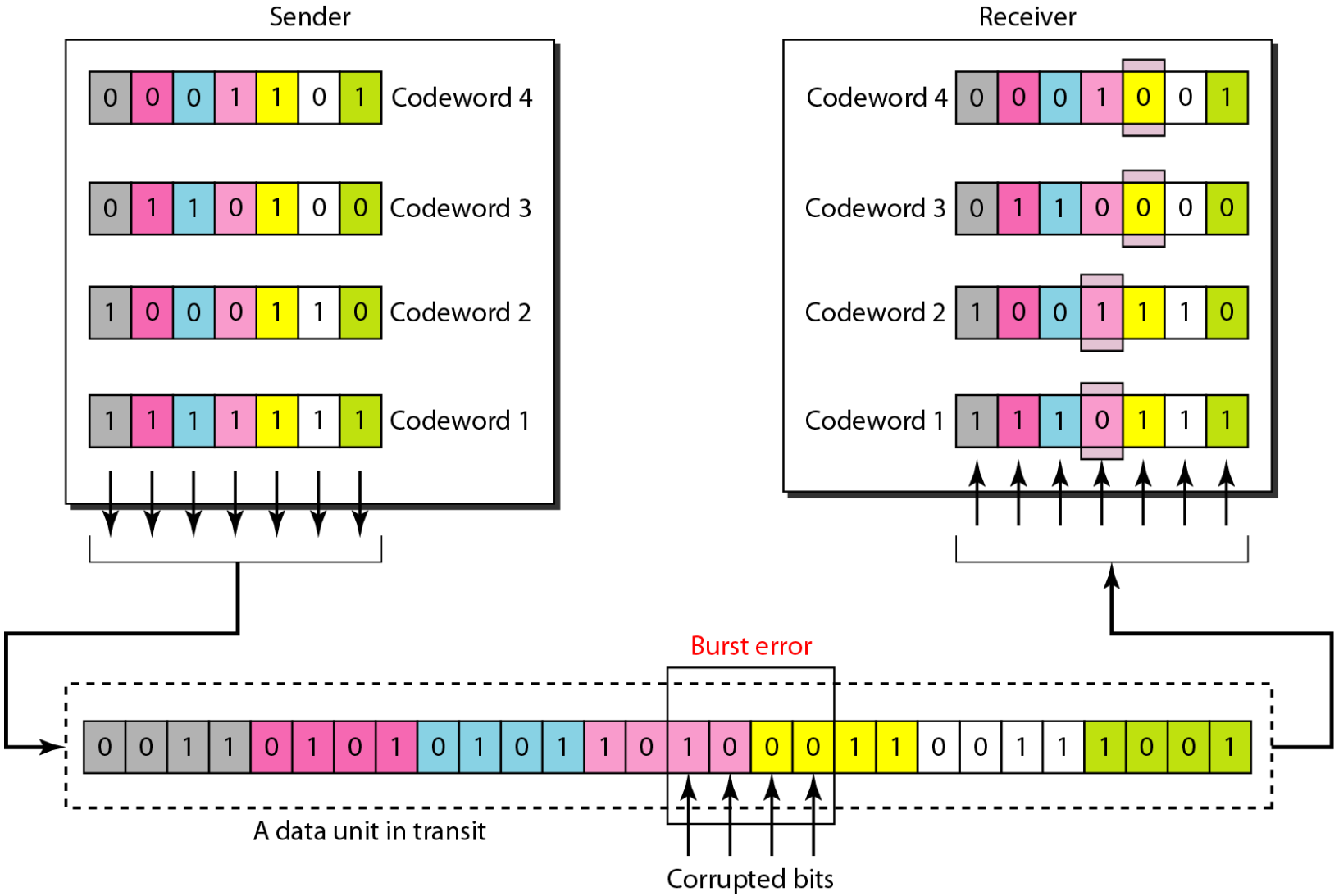
We need to make $k = n - m$ greater than or equal to 7, or $2^m - 1 - m \geq 7$.

1. *If we set $m = 3$, the result is $n = 2^3 - 1$ and $k = 7 - 3$, or 4, which is not acceptable.*
2. *If we set $m = 4$, then $n = 2^4 - 1 = 15$ and $k = 15 - 4 = 11$, which satisfies the condition. So the code is*

C(15, 11)

Figure 10.13 *Burst error correction using Hamming code*

使用交织编码技术的汉明码，提高突发错误能力



10-4 CYCLIC CODES 循环码

***Cyclic codes** are special linear block codes with one extra property. In a cyclic code, if a codeword is cyclically shifted (rotated), the result is another codeword.*

循环码是有一个附加性质的特殊的线性块码。这个性质是如果码字循环移位（旋转），结果还是另一个循环码字。

Topics discussed in this section:

Cyclic Redundancy Check 循环冗余校验码
Hardware Implementation 硬件实现
Polynomials 多项式
Cyclic Code Analysis 循环码性能分析
Advantages of Cyclic Codes 循环码的优点
Other Cyclic Codes 其它循环码

Table 10.6 *A CRC code with C(7, 4)* 循环冗余校验码简称CRC

Dataword	*Codeword*	*Dataword*	*Codeword*
0000	0000000	1000	1000101
0001	0001011	1001	1001110
0010	0010110	1010	1010011
0011	0011101	1011	1011000
0100	0100111	1100	1100010
0101	0101100	1101	1101001
0110	0110001	1110	1110100
0111	0111010	1111	1111111

Figure 10.14 *CRC encoder and decoder*

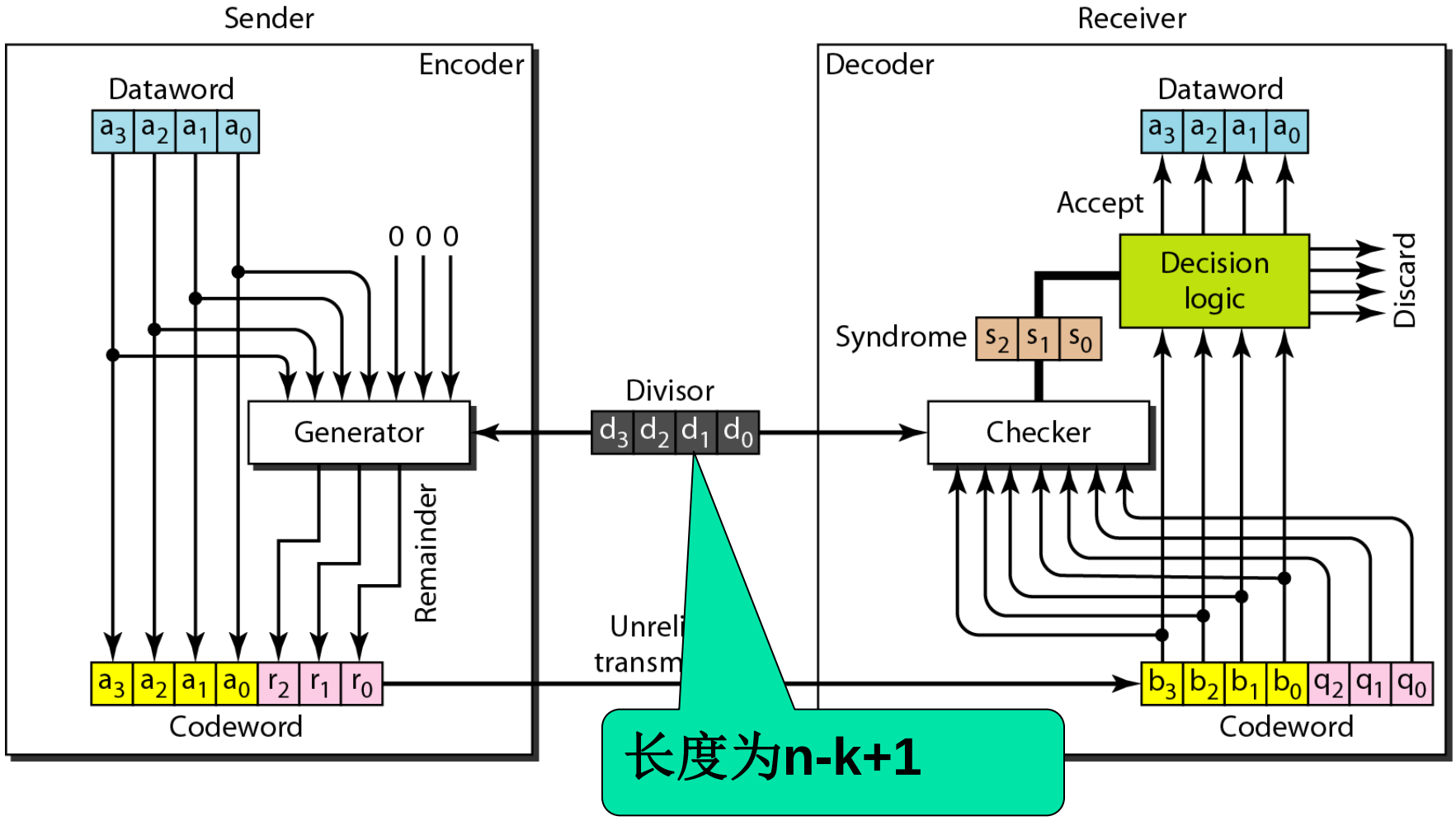


Figure 10.15 *Division in CRC encoder* CRC编码器中的除法（模2运算）

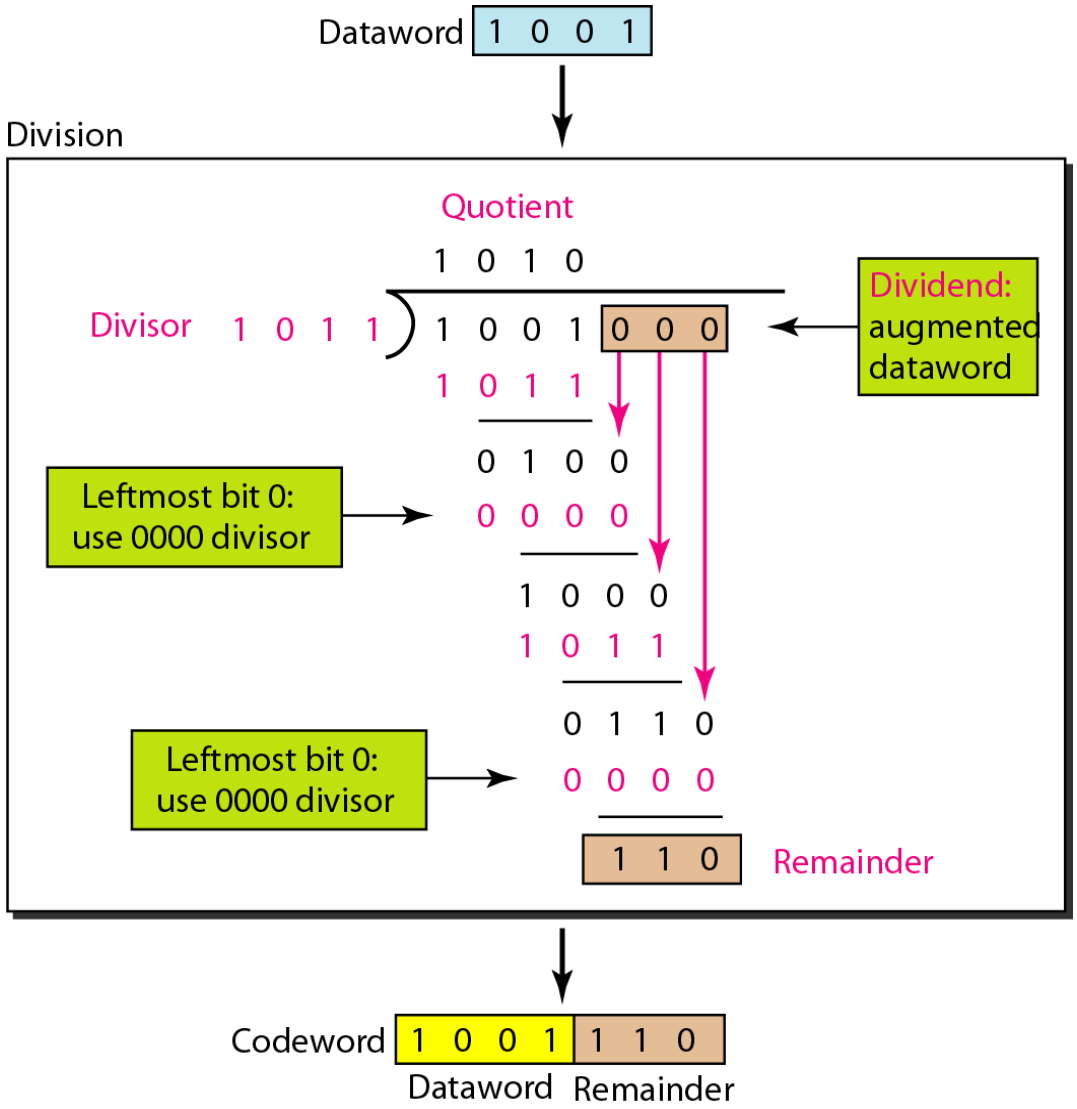


Figure 10.16 ***Division in the CRC decoder for two cases***

CRC的译码器中的除法（除尽和除不尽）

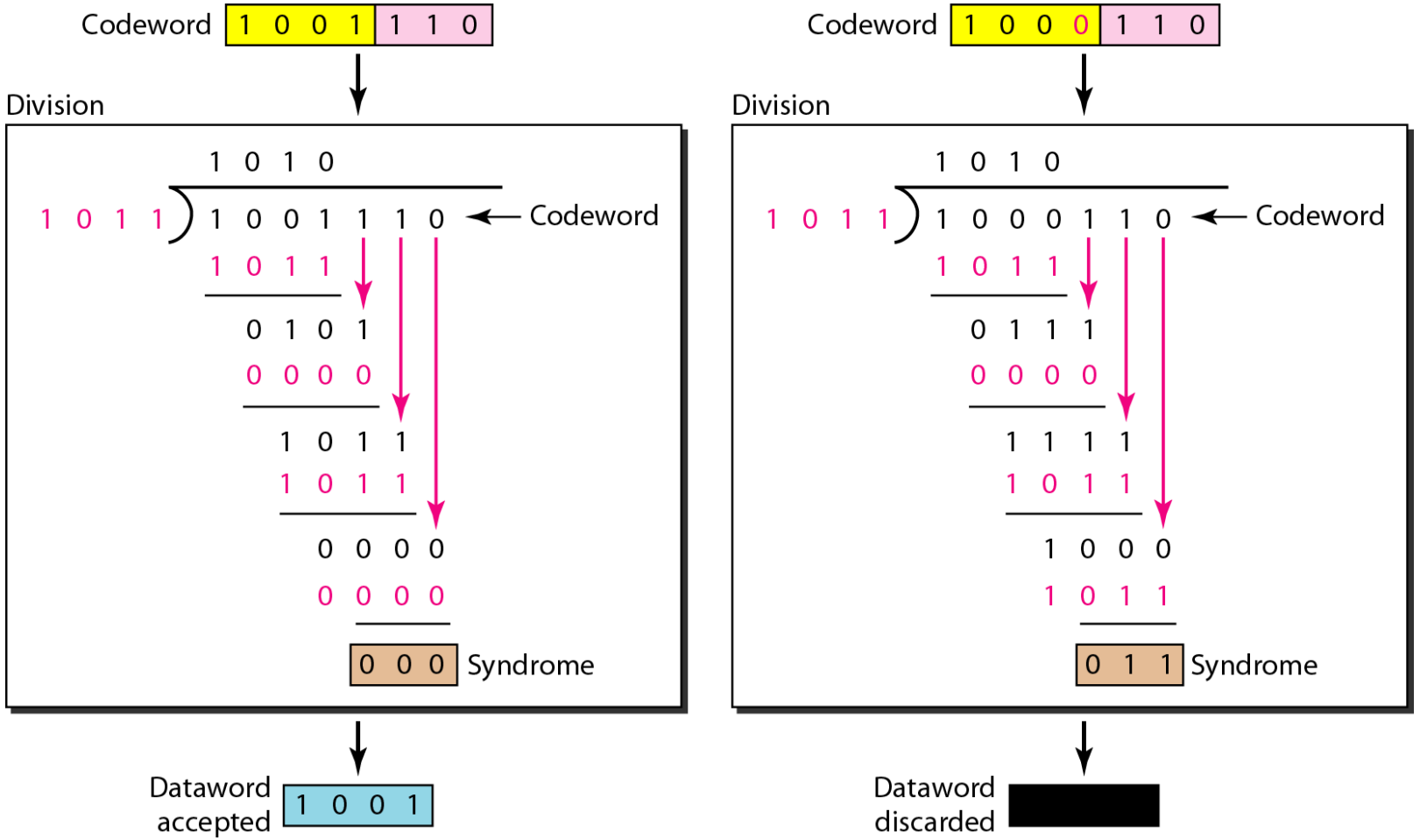


Figure 10.17 *Hardwired design of the divisor in CRC*

CRC的硬件实现不做要求，选讲

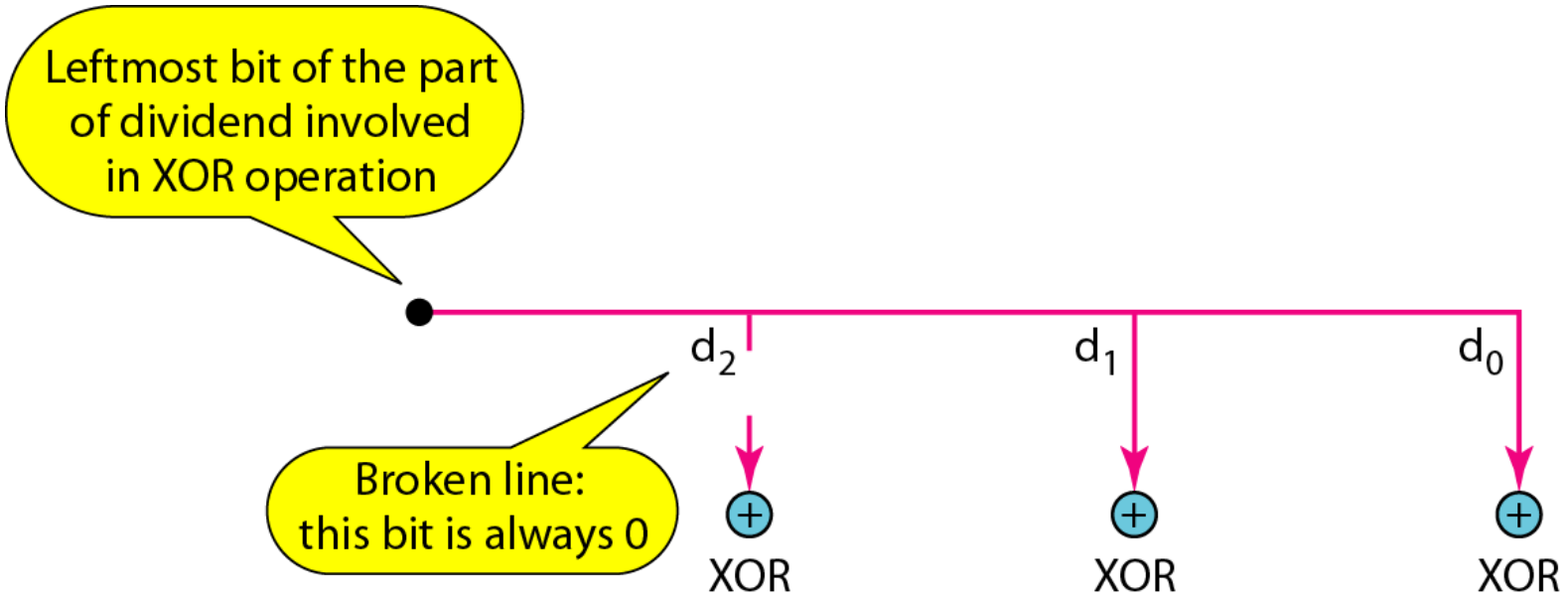


Figure 10.18 *Simulation of division in CRC encoder*

CRC的硬件实现不做要求，选讲

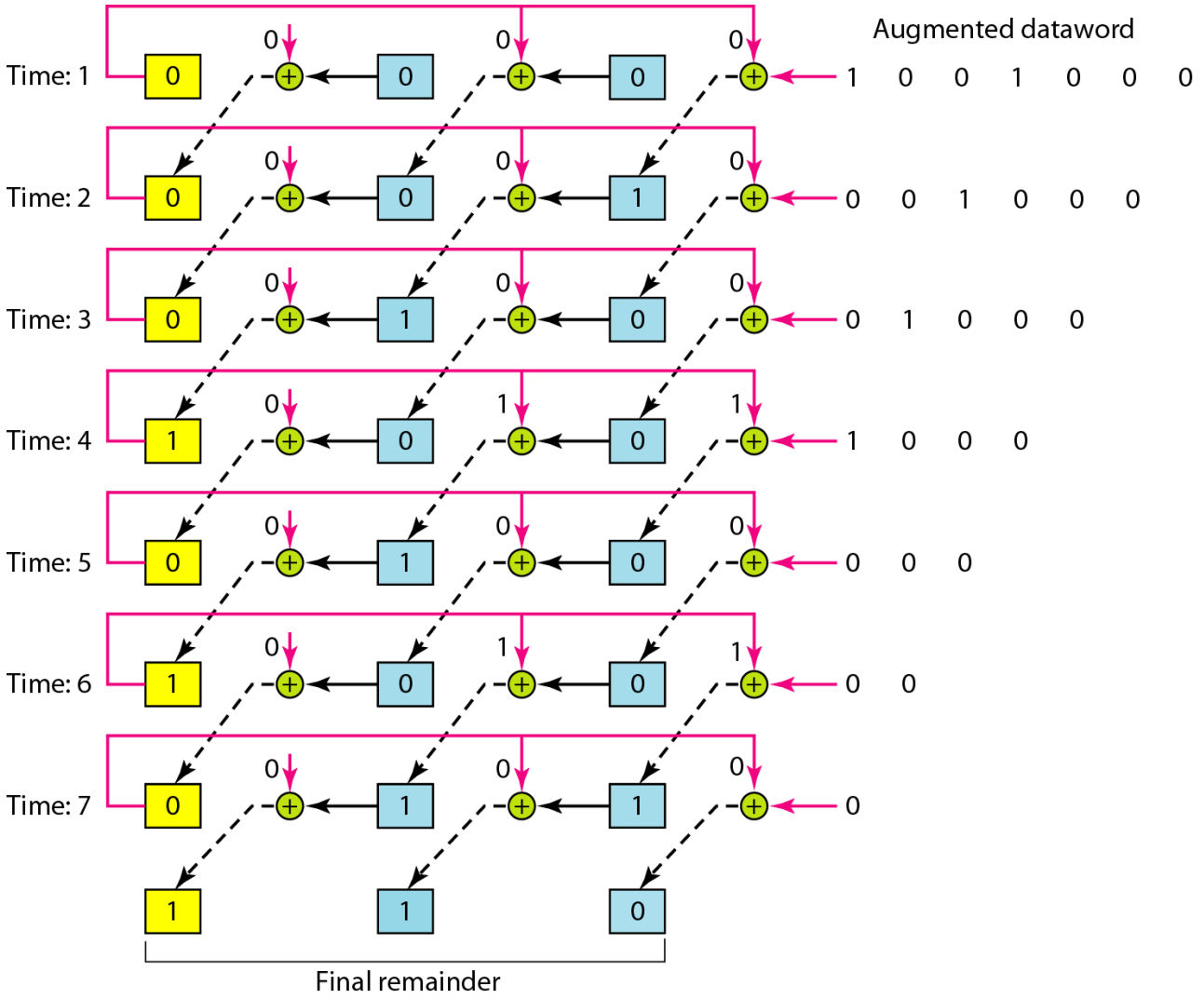


Figure 10.19 *The CRC encoder design using shift registers*

CRC的硬件实现不做要求，选讲

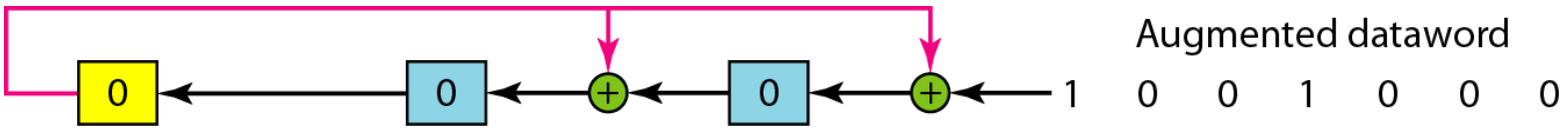


Figure 10.20 *General design of encoder and decoder of a CRC code*

CRC的硬件实现不做要求，选讲

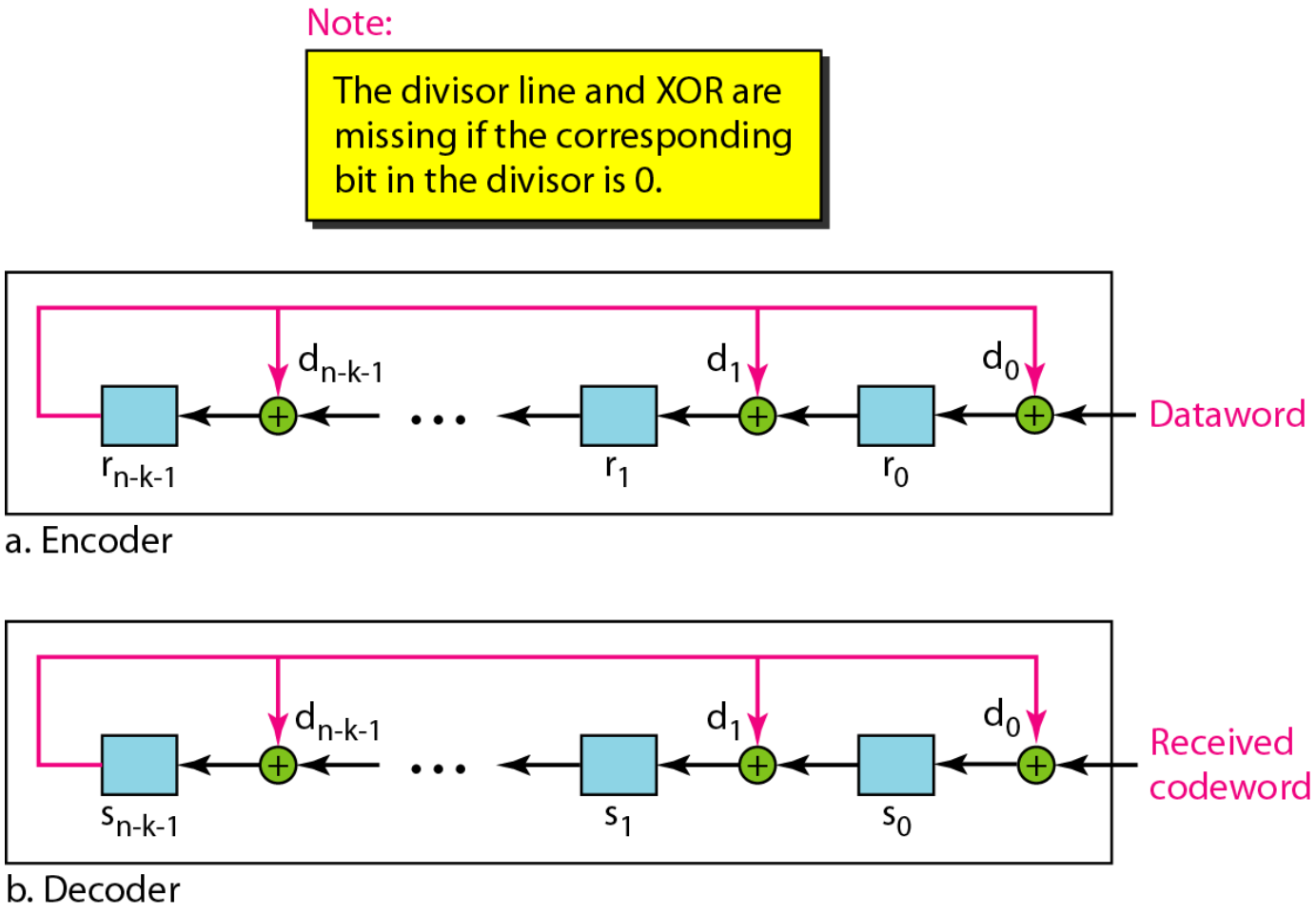
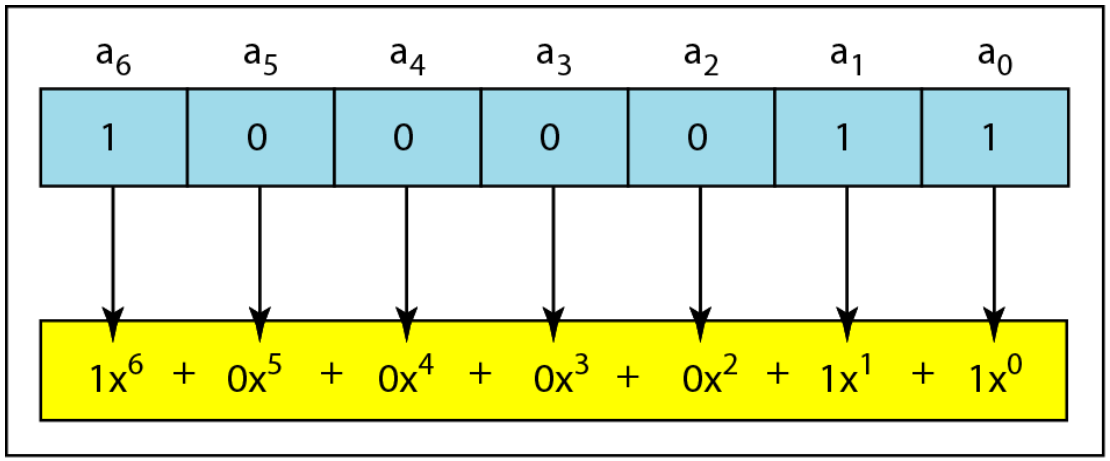
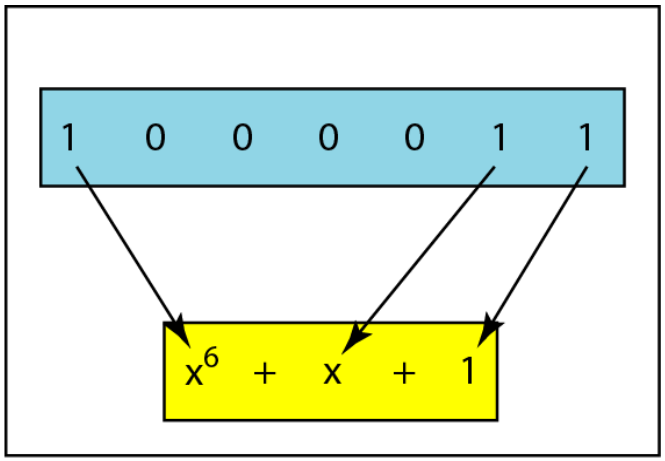


Figure 10.21 *A polynomial to represent a binary word*

循环冗余校验码可以用多项式表示



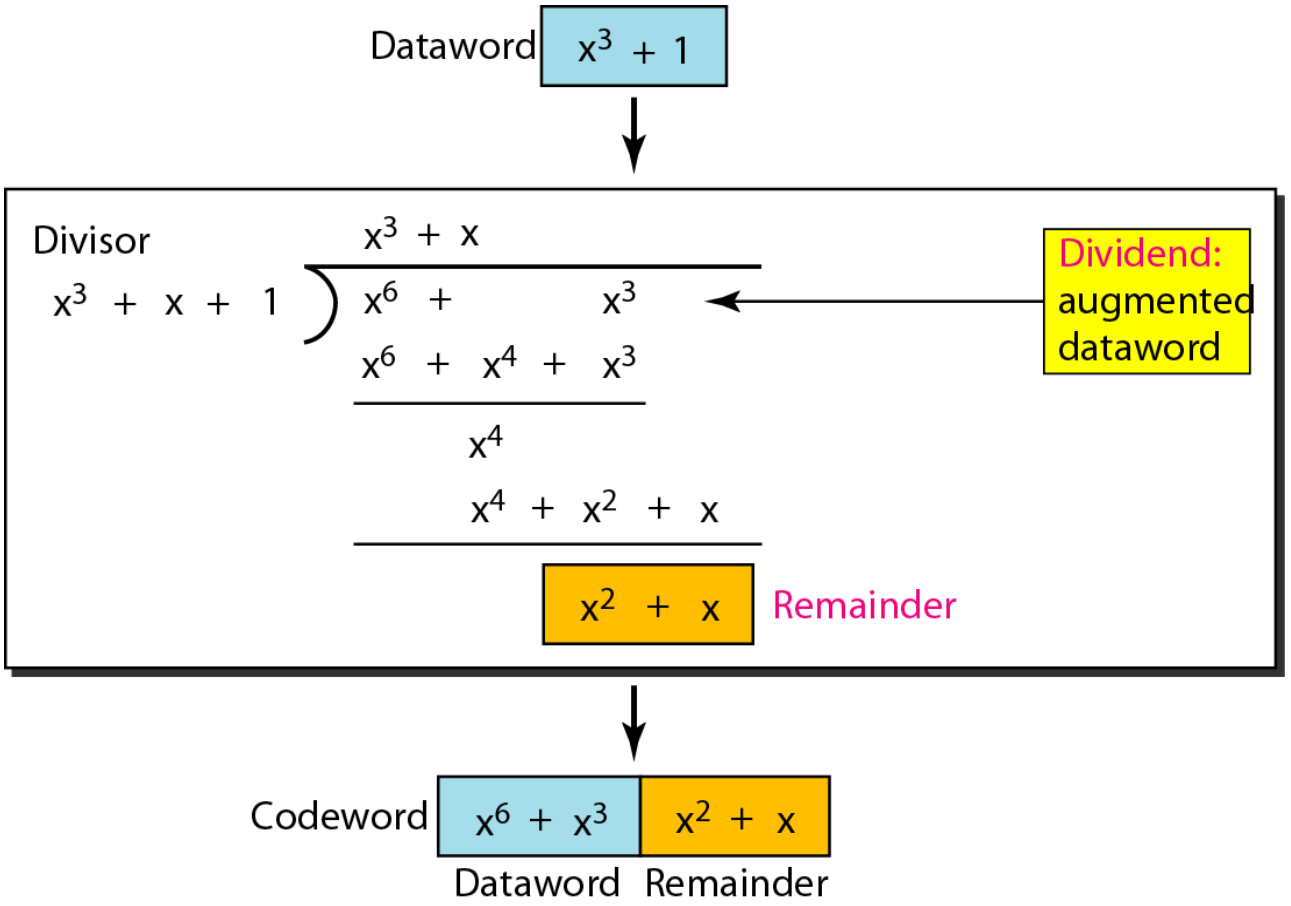
a. Binary pattern and polynomial



b. Short form

Figure 10.22 *CRC division using polynomials*

这样，CRC除法就变成多项式除法



Note

The divisor in a cyclic code is normally called the generator polynomial or simply the generator.

循环码的除数通常称为生成多项式，简称生成子（生成器）

Note

In a cyclic code,

If s(x) ≠ 0, one or more bits is corrupted.

If s(x) = 0, either

a. No bit is corrupted. or

b. Some bits are corrupted, but the decoder failed to detect them.

CRC中，校正子S(x) ≠ 0说明有差错；=0说明无差错或者有差错但译码器无法检测出(超出检错能力)。

Note

In a cyclic code, those *e*(x) errors that are divisible by *g*(x) are not caught.

循环码中，有些可以被生成多项式g(x)整除的差错无法被捕捉到。(发生的概率极小)

单个位差错

Note

If the generator has more than one term and the coefficient of x^0 is 1, all single errors can be caught.

若生成多项式至少有两项，且x^0 的系数是1，则所有单比特错误都可以检出。

Example 10.15

下面三个生成多项式的纠错能力如何？

Which of the following g(x) values guarantees that a single-bit error is caught? For each case, what is the error that cannot be caught?

a. $x + 1$ ***b.*** x^3 ***c.*** 1

Solution

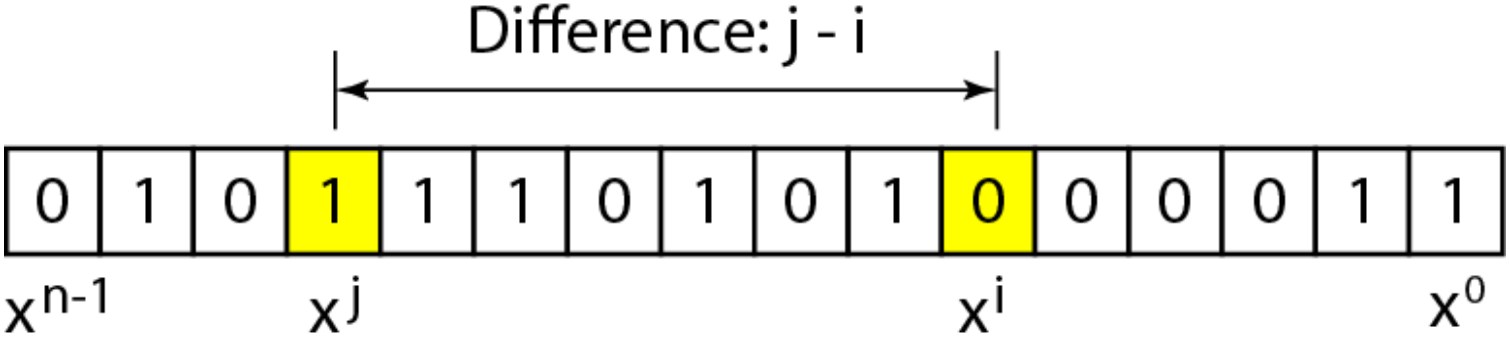
a. *No* x^i *can be divisible by* $x + 1$*. Any single-bit error can be caught.*

b. *If i is equal to or greater than 3,* x^i *is divisible by g(x). All single-bit errors in positions 1 to 3 are caught.*

c. *All values of i make* x^i *divisible by g(x). No single-bit error can be caught. This g(x) is useless.*

Figure 10.23 ***Representation of two isolated single-bit errors using polynomials***
使用多项式表示的两个独立的单比特差错

两个独立的单个位差错



$$x^j + x^i \qquad e(x) = x^j(x^{j-i} + 1)$$

两个独立的单个位差错

Note

If a generator cannot divide $x^t + 1$ (t between 0 and n – 1), then all isolated double errors can be detected.

若生成多项式不能整除$x^t + 1$，那么所有独立的双比特错误都能被检测到。

Example 10.16

下面几个生成多项式对独立的双比特差错能力如何

Find the status of the following generators related to two isolated, single-bit errors.

a. $x + 1$ *b.* $x^4 + 1$ *c.* $x^7 + x^6 + 1$ *d.* $x^{15} + x^{14} + 1$

Solution

a. This is a very poor choice for a generator. Any two errors next to each other cannot be detected.

b. 这个生成多项式不能检测相隔4个位置的两个差错。这两个差错都位于任何位置，但是如果他们的距离是4仍然无法被检测到。

c. This is a good choice for this purpose.

d. 如果t小于32768，这个多项式不能正处类型为$x^t + 1$的任何差错。这表示一个码字中两个对立的差错相邻或者最多离开32768位都能被这个生成多项式检测到。

奇数个差错

Note

A generator that contains a factor of $x + 1$ can detect all odd-numbered errors.

包含$x+1$因子的生成多项式能检测到所有奇数个比特错误。

突发性差错

Note

- All burst errors with $L \leq r$ will be detected.
- All burst errors with $L = r + 1$ will be detected with probability $1 - (1/2)^{r-1}$.
- All burst errors with $L > r + 1$ will be detected with probability $1 - (1/2)^{r}$.

生成多项式纠突发错误的能力

Example 10.17

下面的生成多项式纠突发错误的能力如何

Find the suitability of the following generators in relation to burst errors of different lengths.

a. $x^6 + 1$ *b.* $x^{18} + x^7 + x + 1$ *c.* $x^{32} + x^{23} + x^7 + 1$

Solution

a. This generator can detect all burst errors with a length less than or equal to 6 bits; 3 out of 100 burst errors with length 7 will slip by; 16 out of 1000 burst errors of length 8 or more will slip by.

Example 10.17 (continued)

b. ***This generator can detect all burst errors with a length less than or equal to 18 bits; 8 out of 1 million burst errors with length 19 will slip by; 4 out of 1 million burst errors of length 20 or more will slip by.***

c. ***This generator can detect all burst errors with a length less than or equal to 32 bits; 5 out of 10 billion burst errors with length 33 will slip by; 3 out of 10 billion burst errors of length 34 or more will slip by.***

Note

生成多项式检错能力总结：

A good polynomial generator needs to have the following characteristics:

1. **It should have at least two terms.** 两项
2. **The coefficient of the term x^0 should be 1.** 即x^0系数一定不为0
3. **It should not divide $x^t + 1$, for t between 2 and $n - 1$.** 没有因子$x^t + 1$
4. **It should have the factor $x + 1$.**待讨论?
5. 一定是一个素的生成多项式。（新加的）

Table 10.7 ***Standard polynomials*** 一些常用的标准生成多项式

Name	*Polynomial*	*Application*
CRC-8	$x^8 + x^2 + x + 1$	ATM header
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$	ATM AAL
CRC-16	$x^{16} + x^{12} + x^5 + 1$	HDLC
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	LANs

10-5 CHECKSUM 校验和与反码

最后，介绍一下用校验和进行简单的检错方法。常用于Internet的其它高层协议中。

The last error detection method we discuss here is called the checksum. The checksum is used in the Internet by several protocols although not at the data link layer. However, we briefly discuss it here to complete our discussion on error checking

Topics discussed in this section:

Idea 概念

One's Complement 反码

Internet Checksum 因特网中的校验和

Example 10.18

Suppose our data is a list of five 4-bit numbers that we want to send to a destination. In addition to sending these numbers, we send the sum of the numbers. For example, if the set of numbers is (7, 11, 12, 0, 6), we send (7, 11, 12, 0, 6, 36), where 36 is the sum of the original numbers. The receiver adds the five numbers and compares the result with the sum. If the two are the same, the receiver assumes no error, accepts the five numbers, and discards the sum. Otherwise, there is an error somewhere and the data are not accepted.

发送数据时，同时还发送它们的和（用于检错校验，因此称为校验和）

Example 10.19

We can make the job of the receiver easier if we send the negative (complement) of the sum, called the checksum. In this case, we send (7, 11, 12, 0, 6, −36). The receiver can add all the numbers received (including the checksum). If the result is 0, it assumes no error; otherwise, there is an error.

也可以发送和的补码，这样接收器处理更简单些。

Example 10.20

用反码来解决数据段的进位和借位问题。

How can we represent the number 21 in one's complement arithmetic using only four bits?

Solution

The number 21 in binary is 10101 (it needs five bits). We can wrap the leftmost bit and add it to the four rightmost bits. We have (0101 + 1) = 0110 or 6.

Example 10.21

用反码解决负数问题

How can we represent the number −6 in one's complement arithmetic using only four bits?

Solution

In one's complement arithmetic, the negative or complement of a number is found by inverting all bits. Positive 6 is 0110; negative 6 is 1001. If we consider only unsigned numbers, this is 9. In other words, the complement of 6 is 9. Another way to find the complement of a number in one's complement arithmetic is to subtract the number from $2^n - 1$ (16 − 1 in this case).

Example 10.22

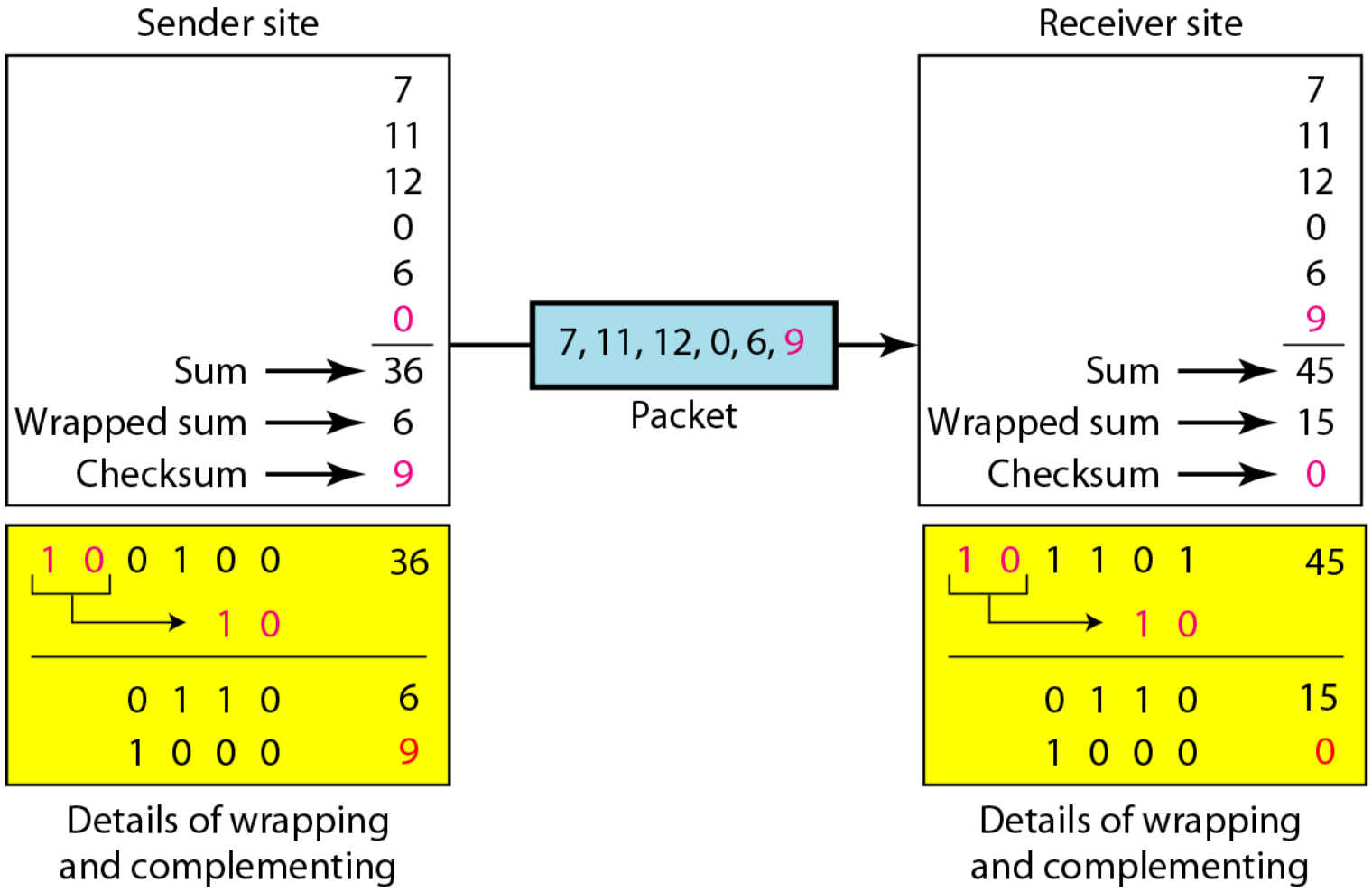
用后面图来解释此例子

Let us redo Exercise 10.19 using one's complement arithmetic. Figure 10.24 shows the process at the sender and at the receiver. The sender initializes the checksum to 0 and adds all data items and the checksum (the checksum is considered as one data item and is shown in color). The result is 36. However, 36 cannot be expressed in 4 bits. The extra two bits are wrapped and added with the sum to create the wrapped sum value 6. In the figure, we have shown the details in binary. The sum is then complemented, resulting in the checksum value 9 (15 − 6 = 9). The sender now sends six data items to the receiver including the checksum 9.

Example 10.22 (continued)

The receiver follows the same procedure as the sender. It adds all data items (including the checksum); the result is 45. The sum is wrapped and becomes 15. The wrapped sum is complemented and becomes 0. Since the value of the checksum is 0, this means that the data is not corrupted. The receiver drops the checksum and keeps the other data items. If the checksum is not zero, the entire packet is dropped.

Figure 10.24 *Example 10.22* 注意分段，进位，和反码等相关的操作
(注意: 图左右下角有错)



Note

因特网校验和的步骤： (发送方)

Sender site:

1. The message is divided into 16-bit words.
2. The value of the checksum word is set to 0.
3. All words including the checksum are added using one's complement addition. (使用反码运算相加)
4. The sum is complemented and becomes the checksum. (累加和求反码变成校验和)
5. The checksum is sent with the data.

Note

因特网校验和的步骤： (接收方)

Receiver site:

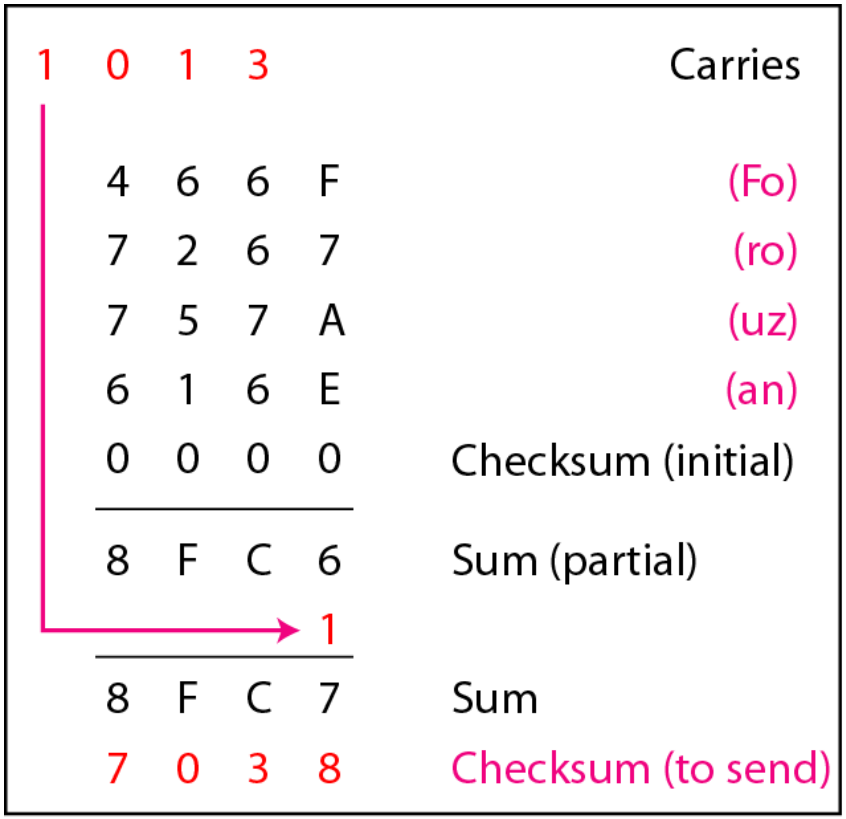
1. **The message (including checksum) is divided into 16-bit words.**
2. **All words are added using one's complement addition. (使用反码运算相加)**
3. **The sum is complemented and becomes the new checksum. (求反码变成新校验和)**
4. **If the value of checksum is 0, the message is accepted; otherwise, it is rejected.**

Example 10.23

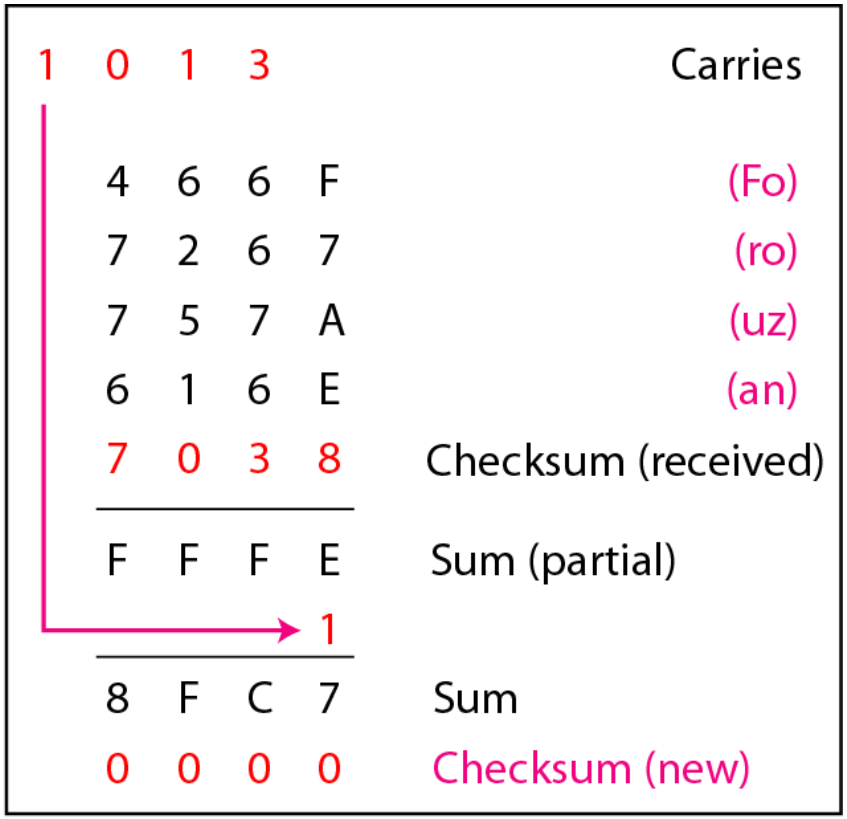
用后面图来解释此例子

Let us calculate the checksum for a text of 8 characters ("Forouzan"). The text needs to be divided into 2-byte (16-bit) words. We use ASCII (see Appendix A) to change each byte to a 2-digit hexadecimal number. For example, F is represented as 0x46 and o is represented as 0x6F. Figure 10.25 shows how the checksum is calculated at the sender and receiver sites. In part a of the figure, the value of partial sum for the first column is 0x36. We keep the rightmost digit (6) and insert the leftmost digit (3) as the carry in the second column. The process is repeated for each column. Note that if there is any corruption, the checksum recalculated by the receiver is not all 0s. We leave this an exercise.

Figure 10.25 *Example 10.23* 注意分段，进位，和反码等相关的操作 (注意: 图右下角有错)



a. Checksum at the sender site



a. Checksum at the receiver site

本章作业：

1，4，7，8，
15，16，18，23，24，26，27，28，
30，32